

INFORMATICA

Clase de hoy

- Formato para lectura-escritura
- Archivos

- Escritura con formato

```
character(8)      :: cadena  
real              :: z  
integer          :: n
```

```
cadena = 'ejemplo '  
z = 3.14159  
n = 112358
```

```
write(*,*) cadena,'n = ',n,'z (decimal) = ',z,'z (exponencial) = ',z
```


• Escritura con formato

```
character(8)      :: cadena
real              :: z
integer           :: n
```

```
cadena = 'ejemplo '
z = 3.14159
n = 112358
```

```
write(*,*) cadena,'n = ',n,'z (decimal) = ',z, 'z (exponencial) = ',z
```

```
ejemplo:n =      112358z <decimal> =      3.14159    z <exponencial> =      3.14159
Press RETURN to close window . . .
```

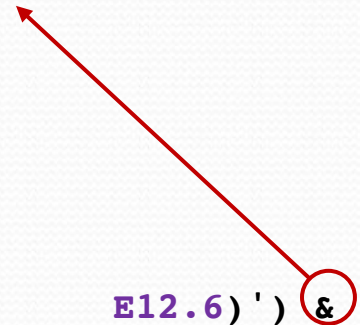
• Escritura con formato

```
character(8)      :: cadena
real              :: z
integer           :: n
```

```
cadena = 'ejemplo '
z = 3.14159
n = 112358
```

```
Write (*,fmt='(a8, 1X,a4,1X,i6,1X,a14,1X,      F8.6,1X,a18,1X,      E12.6)') &
      cadena, 'n = ',n,      'z (decimal) = ',z,      'z (exponencial) = ',z
```

Símbolo de continuación:
La sentencia continúa en la
línea siguiente



```
ejemplo: n = 112358 z (decimal) = 3.141590 z (exponencial) = 0.314159E+01
Press RETURN to close window . . .
```


• Escritura con formato

```
...  
Write (*,fmt='(a8,1X,a4,1X,i6,1X,a14,1X,F8.6,1X,a18,1X,E12.6)') &  
cadena,'n = ',n,'z (decimal) = ',z,'z (exponencial) = ',z
```

`fmt='(f1,f2,..)'` : lista de formatos con los que se van a leer o escribir los datos.

Opciones: `In[m]` I implica integer

n número de espacios reservados

m número de dígitos de la representación [opcional].

`Fn.d` F implica real

n número de espacios reservados

d número de dígitos decimales.

`En.d` E implica real en representación exponencial

n número de espacios reservados

d número de dígitos decimales.

• Escritura con formato

```
...  
Write (*,fmt='(a8,1X,a4,1X,i6,1X,a14,1X,F8.6,1X,a18,1X,E12.6)') &  
cadena,'n = ',n,'z (decimal) = ',z,'z (exponencial) = ',z
```

`fmt='(f1,f2,..)'` : lista de formatos con los que se van a leer o escribir los datos.

Opciones: `A[n]` A implica character

`n` número de espacios reservados.

`nX` X implica espacio en blanco

`n` número de espacios en blanco.

Si no se quiere utilizar formato basta con usar *

Si se va a utilizar varias veces el mismo formato se puede definir previamente

```
character(len=*), PARAMETER :: FMT1 =&  
    '(a8,1X,a4,1X,i6,1X,a14,1X,F8.6,1X,a18,1X,E12.6)'
```

```
write(*,FMT1) ... variables y expresiones ...
```

0

```
character(50) :: FMT2
```

```
FMT2 = '(a8,1X,a4,1X,i6,1X,a14,1X,F8.6,1X,a18,1X,E12.6)'
```

```
write(*,FMT2) ... variables y expresiones ...
```


• Ejemplo

```
program formatos
```

```
implicit none
```

```
!Declaracion de variables
```

```
integer                :: n,i
```

```
integer,allocatable    :: U(:)
```

```
real    ,allocatable    :: X(:)
```

```
real    ,parameter     :: pi = acos(-1.)
```

```
!Cuerpo del programa
```

```
n=5
```

```
allocate(X(n),U(n))
```

```
do i = 1, n
```

```
    U(i) = 2*i + 1
```

```
    X(i) = cos((i-1)*pi/(n-1))
```

```
enddo
```

```
!Escribir los vectores
```

```
write(*,fmt='(5(2x,I4))')    U
```

```
write(*,*)    ! Escribe linea en blanco
```

```
write(*,fmt='(5(2x,f10.5))') X
```

```
write(*,*)    ! Escribe linea en blanco
```

```
write(*,fmt='(5(2x,E10.4))') X
```

```
endprogram
```

• Ejemplo

```
program formatos
```

```
implicit none
```

```
!Declaracion de variables
```

```
integer                :: n,i
```

```
integer,allocatable   :: U(:)
```

```
real    ,allocatable   :: X(:)
```

```
real    ,parameter    :: pi = acos(-1.)
```

```
!Cuerpo del programa
```

```
n=5
```

```
allocate(X(n),U(n))
```

```
do i = 1, n
```

REPITE 5 VECES EL FORMATO DEL PARENTESIS

```
    U(i) = 2*i +
```

```
    X(i) = cos((i-1)*pi/(n-1))
```

```
enddo
```

```
!Escribir los valores
```

```
write(*,fmt='(5(2x,I4))')      U
```

```
write(*,*)                    ! Escribe linea en blanco
```

```
write(*,fmt='(5(2x,f10.5))') X
```

```
write(*,*)                    ! Escribe linea en blanco
```

```
write(*,fmt='(5(2x,E10.4))') X
```

```
endprogram
```


• Ejemplo

```
program formatos
```

```
implicit none
```

```
!Declaracion de variables
```

```
integer      :: n,i  
integer,allocatable :: U(:)  
real    ,allocatable :: X(:)  
real    ,parameter  :: pi = acos(-1.)
```

```
!Cuerpo del programa
```

```
n=5
```

```
allocate(X(n),U(n))
```

```
do i = 1, n  
  U(i) = 2*i + 1  
  X(i) = cos((i-1)*pi/(n-1))  
enddo
```

```
!Escribir los vectores
```

```
write(*,fmt='(5(2x,I4))') U  
write(*,*)      ! Escribe linea en blanco  
write(*,fmt='(5(2x,f10.5))') X  
write(*,*)      ! Escribe linea en blanco  
write(*,fmt='(5(2x,E10.4))') X
```

```
endprogram
```

```
      3      5      7      9     11  
      1.00000      0.70711     -0.00000     -0.70711     -1.00000  
      0.1000E+01  0.7071E+00  -.4371E-07  -.7071E+00  -.1000E+01  
Presione una tecla para continuar . . . _
```


• Escritura/Lectura en ficheros

```
character(8)      :: cadena
character         :: basura
real              :: z
integer           :: n
```

```
cadena = 'ejemplo '
z = 3.14159
n = 112358
```

```
open(unit = 10 ,file = 'prueba.dat')
```

```
write(10, fmt='(a8,1X,a4,1X,i6,1X,a14,1X,F8.6,1X,a18,1X,E12.6)') &
    cadena, 'n = ', n, 'z (decimal) = ', z, 'z (exponencial) = ', z
```

```
rewind(unit=10)
```

```
read(10, fmt = '(a8,1X,a4,1X,i6,1X,a14,1X,F8.6,1X,a18,1X,E12.6)')
    cadena, basura, n, basura, z, basura, z
```

```
close(10)
```

• Escritura/Lectura en ficheros

```
open(unit = 10 ,file = 'prueba.dat')
```

```
open(unit=u, file='hola.dat', status='new', iostat=ierr )
```

<code>unit=u</code>	: <code>u</code> es la unidad de referencia para el fichero abierto.
<code>file=name</code>	: <code>name</code> dato de tipo character con el nombre del fichero
[opcional] <code>status='new'</code>	: El fichero no existía previamente.
<code>'old'</code>	: El fichero ya existe
<code>'unknown'</code>	: No sabemos si existe. Si no existe lo crea (por defecto)
<code>'scratch'</code>	: Crea el fichero y lo destruye al acabar la ejecución.
[opcional] <code>action='read'</code>	: El fichero se usará para leer datos.
<code>'write'</code>	: El fichero se usará para escribir.
<code>'readwrite'</code>	: El fichero se puede usar para ambas (por defecto).
[opcional] <code>iostat=info</code>	: <code>info</code> variable entera que devuelve 0 si no ha habido error.

- Escritura/Lectura en ficheros

```
write(10, fmt='(a8,1X,a4,1X,i6,1X,a14,1X,F8.6,1X,a18,1X,E12.6)') &  
cadena,'n = ',n,'z (decimal) = ',z,'z (exponencial) = ',z
```

```
write(unit=u, fmt='(f1,f2,...)', iostat=ierr ) datos
```

```
read(unit=u, fmt='(f1,f2,...)', iostat=ierr ) datos
```

unit=u : u unidad del fichero del que se va a leer o en el que se escribe.

Si se va a leer del teclado o escribir en pantalla se usa *

fmt='(f1,f2,..)' : lista de formatos con los que se van a leer o escribir los datos.

[opcional] **iostat=info** : **info** variable entera que devuelve:

0 Si no ha habido error.

<0 Si alcanza el final del fichero durante la lectura.

>0 Si detecta error durante la lectura o escritura.

datos Conjunto de variables, separadas por comas, que se van a leer o escribir.

Ej. **write(u,*)** Escribe una línea en blanco

read(u,*) Salta de línea en durante la lectura.

- Escritura/Lectura en ficheros

`rewind(unit=10)`



`rewind(unit=u)` Sitúa el cursor al inicio del fichero a leer o escribir

`close(10)`

`close(unit=u)`

`unit=u` : Cierra el fichero que se abrió en la unidad `u`.

```

1 program formatos
2
3 implicit none
4
5 !Declaracion de variables
6 integer :: n,i
7 real ,allocatable :: X(:)
8 real ,parameter :: pi = acos(-1.)
9
10 !Cuerpo del programa
11 n=5
12 allocate(X(n))
13
14 do i = 1, n
15   X(i) = cos((i-1)*pi/(n-1))
16 enddo
17
18 open (unit= 10, file='fichero.dat')
19
20 !Escribir los vectores
21
22 write(10,*)      ! Escribe linea en blanco
23 write(10,fmt='(5(2x,f10.5))') X
24 write(*,*)      ! Escribe linea en blanco
25
26 X = 0.
27
28 rewind(10)
29
30 read(10,*)      ! Lee linea en blanco
31 read(10,fmt='(5(2x,f10.5))') X
32 read(10,*)      ! Lee linea en blanco
33
34 write(*,fmt='(5(2x,f10.5))') X
35
36 end program

```

```
1 program formatos
```

```
2  
3 implicit none
```

```
4  
5 !Declaracion de variables
```

```
6 integer
```

```
7 real, allocatable
```

```
8 real, parameter
```

```
9  
10 !Cuerpo del programa
```

```
11 n=5
```

```
12 allocate(X(n))
```

```
13  
14 do i = 1, n
```

```
15 X(i) = cos((i-1)*pi/
```

```
16 enddo
```

```
17  
18 open (unit= 10, file=
```

```
19  
20 !Escribir los vectores
```

```
21  
22 write(10,*) ! Escribe linea en blanco
```

```
23 write(10,fmt='(5(2x,f10.5))') X
```

```
24 write(*,*) ! Escribe linea en blanco
```

```
25  
26 X = 0.
```

```
27  
28 rewind(10)
```

```
29  
30 read(10,*) ! Lee linea en blanco
```

```
31 read(10,fmt='(5(2x,f10.5))') X
```

```
32 read(10,*) ! Lee linea en blanco
```

```
33  
34 write(*,fmt='(5(2x,f10.5))') X
```

```
35  
36 end program
```

fichero.dat

1

2

3

1.000000

0.70711

-0.000000

-0.70711

-1.000000

C:\Windows\system32\cmd.exe

1.000000

0.70711

-0.000000

-0.70711

-1.000000

Presione una tecla para continuar . . .


```

1 program formatos
2
3 implicit none
4
5 !Declaracion de variables
6 integer :: n,i
7 real ,allocatable :: X(:)
8 real ,parameter :: pi = acos(-1.)
9
10 !Cuerpo del programa
11 n=5
12 allocate(X(n))
13
14 do i = 1, n
15   X(i) = cos((i-1)*pi/(n-1))
16 enddo
17
18 open (unit= 10, file='fichero.dat')
19
20 !Escribir los vectores
21
22 write(10,*) ! Escribe linea en blanco
23 write(10,fmt='(5(2x,f10.5))') X
24 write(10,*) ! Escribe linea en blanco
25
26 X = 0.
27
28 rewind(10)
29
30 read(10,*) ! Lee linea en blanco
31 read(10,*) X
32 read(10,*) ! Lee linea en blanco
33
34 write(*,fmt='(5(2x,f10.5))') X
35
36 end program

```

fichero.dat

```

1
2      1.00000      0.70711      -0.00000      -0.70711      -1.00000
3
4

```

C:\Windows\system32\cmd.exe

```

      1.00000      0.70711      -0.00000      -0.70711      -1.00000
Presione una tecla para continuar . . . _

```

Practica

```
program leer_y_escribir

implicit none
! Declaracion de variables
integer :: i,a,ierr
real    :: b

open(unit=12,file='hola.dat',status='new', iostat=ierr )
write(*,*) ierr

do i=1,5
    write(*,*) 'Escribe un numero: '
    read(*,*) a
    write(12,fmt='(i4)') a
enddo

rewind(12)

do i=1,5
    read(12,*) b
    write(*,fmt='(f10.4)') b
enddo

close(12, iostat=ierr)

endprogram
```

Copia el siguiente código. La segunda vez que lo ejecutes dará un error ¿Por qué?. Prueba con diferentes formatos de lectura y escritura.

Serie de Taylor: PROGRAMACION (Aclaraciones)

Escribir un programa (versión básica) que:

- Pida por pantalla un entero N y un real x
- Evalúe el desarrollo de Taylor ($p(x)$) de la función $\sin(x)$ con N términos en el punto x
- Escriba por pantalla:
 - El grado del polinomio utilizado (grado \neq N, en general).
 - El valor del desarrollo de Taylor en x: $p(x)$.
 - El valor de la función en x: $\sin(x)$.
 - El error cometido: $|p(x) - \sin(x)|$

NOTA: el polinomio de Taylor centrado en $x=0$
de $f(x)=\sin(x)$ es

$$p(x) = \sum_{n=0}^N a_n x^{2n+1}$$

donde los coeficientes a_n valen:

$$a_n = \frac{(-1)^n}{(2n+1)!}$$

RESULTADOS:

```
N          1 x  1.00000000
p(x)  0.833333313
sin(x) 0.841470957
error   8.13764334E-03
```

```
N          2 x  1.00000000
p(x)  0.841666639
sin(x) 0.841470957
error   1.95682049E-04
```

```
N          3 x  1.00000000
p(x)  0.841468215
sin(x) 0.841470957
error   2.74181366E-06
```


Serie de Taylor: PROGRAMACION (Aclaraciones)


Escribir un programa (versión avanzada) que:

- Pida por pantalla dos reales: x , tol y un entero: N .
- Calcule N términos a_n del desarrollo de Taylor de la función $\sin(x)$ y los almacene en un vector de tamaño N .
- Evalúe el desarrollo de Taylor en el punto x , usando el mínimo número de términos necesarios para que el error sea menor que tol
- Escriba por pantalla:
 - El el valor del desarrollo de Taylor en x .
 - El valor de la función en x .
 - El error cometido $|\text{Taylor}(x) - \sin(x)|$
 - El grado del polinomio utilizado (grado $\neq N$, en general).
 - En caso de que el error cometido sea mayor que tol con el máximo número de términos (N), pedirle al usuario que utilice más términos y finalizar el programa.

NOTA: el polinomio de Taylor centrado en $x=0$ de $f(x)=\sin(x)$ es

$$p(x) = \sum_{n=0}^N a_n x^{2n+1}$$

donde los coeficientes a_n valen:


$$a_n = \frac{(-1)^n}{(2n+1)!}$$

RESULTADOS:

N	2	x	1.00000000
p(x)	0.841666639		
sin(x)	0.841470957		
error	1.95682049E-04		

N	3	x	1.00000000
p(x)	0.841468215		
sin(x)	0.841470957		
error	2.74181366E-06		

Serie de Taylor: PROGRAMACION (Aclaraciones)

Comentarios adicionales:

- Si el cálculo del factorial os devuelve un número negativo, es posible que estéis teniendo problemas de overflow (no hay memoria suficiente reservada para almacenar un entero tan grande).
- Este problema podéis resolverlo almacenando el resultado del factorial en una variable de tipo real.
- Si queréis precisión adicional, es recomendable utilizar variables de doble precisión [real(8)].
- En el programa final, debería ser sencillo cambiar el desarrollo de Taylor para que calcule el de otra función. Por ejemplo, el desarrollo del coseno centrado en $x=0$ es:

$$p(x) = \sum_{n=0}^N a_n x^{2n} \quad a_n = \frac{(-1)^n}{(2n)!}$$

- Mientras que el de la función exponencial, también centrado en $x=0$, es:

$$p(x) = \sum_{n=0}^N a_n x^n \quad a_n = \frac{1}{n!}$$

PRÁCTICA ENTREGABLE II

Trabajo para casa: (deadline 12 Noviembre - 23:55)

- Realizar la práctica de Taylor (versión básica o versión avanzada).
- EVALUACIÓN:
 - Versión básica nota máxima 5/10
 - Versión avanzada nota máxima 8/10
 - Versión básica + formato y escritura a fichero nota máxima 7/10
 - Versión avanzada + formato y escritura a fichero nota máxima 10/10

NOTAS:

- El formato de escritura lo elegís vosotros. Elegid el que consideréis más adecuado.
- Si el código escribe en un fichero, debe escribir también la misma información por pantalla.
- Se valorará positivamente si la información (N, tol, x) se introduce desde un archivo en lugar de por pantalla.
- El Jueves 16 de Noviembre, se realizará un pequeño test en clase para el que tendréis que **modificar** y **ejecutar** el código entregado.
- Para este test será necesario un portátil con conexión a internet y el compilador instalado.
- iiiCOPIAR!!! (HE DETECTADO COPIAS EN LA DE LOS NÚMEROS PRIMOS)

- MATERIAL ADICIONAL

- Ejemplo EXTRA (A priori yo no sé cuantos números voy a escribir «allocatable»)

```
program formatos
```

```
!Declaracion de variables
```

```
integer                :: n,i
```

```
real    ,allocatable :: X(:)
```

```
real    ,parameter   :: pi = acos(-1.)
```

```
! Variables complementaria
```

```
character(2)          :: n_c
```

```
character(40)         :: formato_escritura
```

```
!Cuerpo del programa
```

```
n=5
```

```
allocate(X(n),U(n))
```

```
! Fuera del contenido del curso
```

```
write(n_c,fmt='(I2)') n
```

```
formato_escritura = ' ( '//trim(n_c)//' (2x,f10.5)'//') '
```

```
do i = 1, n
```

```
    X(i) = cos((i-1)*pi/(n-1))
```

```
enddo
```

```
!Escribir los vectores
```

```
write(*,*)      ! Escribe linea en blanco
```

```
write(*,fmt='(5(2x,f10.5))') X
```

```
write(*,*)      ! Escribe linea en blanco
```

```
write(*,fmt=formato_escritura) X
```


- Ejemplo EXTRA (A priori yo no sé cuantos números voy a escribir «allocatable»)

```
program formatos
```

```
!Declaracion de variables
```

```
integer          :: n,i  
real ,allocatable :: X(:)  
real ,parameter  :: pi = acos(-1.)  
! Variables complementaria  
character(2)     :: n_c  
character(40)    :: formato_escritura
```

```
!Cuerpo del programa
```

```
n=5  
allocate(X(n),U(n))
```

```
! Fuera del contenido del curso
```

```
write(n_c,fmt='(I2)') n  
formato_escritura = ' ( '//trim(n_c)//' (2x,f10.5)'//') '
```

```
do i = 1, n  
    X(i) = cos((i-1)*pi/(n-1))  
enddo
```

```
!Escribir los vectores
```

```
write(*,*)      ! Escribe linea en blanco  
write(*,fmt='(5(2x,f10.5))') X  
write(*,*)      ! Escribe linea en blanco  
write(*,fmt=formato_escritura) X
```

TEXTO

GUARDA EN LA VARIABLE **n_c** EL VALOR NÉMERICO CONTENIDO EN n

- Ejemplo EXTRA (A priori yo no sé cuantos números voy a escribir «allocatable»)

```
program formatos
```

```
!Declaracion de variables
```

```
integer                :: n,i  
real    ,allocatable  :: X(:)  
real    ,parameter    :: pi = acos(-1.)  
! Variables complementaria  
character(2)           :: n_c  
character(40)          :: formato_escritura
```

```
!Cuerpo del programa
```

```
n=5  
allocate(X(n),U(n))
```

```
! Fuera del contenido del curso
```

```
write(n_c,fmt='(I2)') n  
formato_escritura = ' ( '//trim(n_c)//' (2x,f10.5)'//') '
```

```
do i = 1, n  
    X(i) = cos((i-1)*pi/(n-1))  
enddo
```

```
!Escribir los vectores
```

```
write(*,*)      ! Escribe linea en blanco  
write(*,fmt='(5(2x,f10.5))') X  
write(*,*)      ! Escribe linea en blanco  
write(*,fmt=formato_escritura) X
```

formato_escritura = (5(2x,f10.5))

- Ejemplo EXTRA (A priori yo no sé cuantos números voy a escribir «allocatable»)

```
program formatos
```

```
!Declaracion de variables
```

```
integer                :: n,i  
real    ,allocatable  :: X(:)  
real    ,parameter    :: pi = acos(-1.)  
! Variables complementaria  
character(2)          :: n_c  
character(40)         :: formato_escritura
```

```
!Cuerpo del programa
```

```
n=5  
allocate(X(n),U(n))  
  
! Fuera del contenido  
write(n_c,fmt='(I2)')  
formato_escritura = '  
  
do i = 1, n  
    X(i) = cos((i-1)*pi  
enddo
```

```
!Escribir los vectores
```

```
write(*,*)      ! Escribe linea en blanco  
write(*,fmt='(5(2x,f10.5))') X  
write(*,*)      ! Escribe linea en blanco  
write(*,fmt=formato_escritura) X
```

```
1.000000    0.70711    -0.00000    -0.70711    -1.00000  
1.000000    0.70711    -0.00000    -0.70711    -1.00000  
Presione una tecla para continuar . . . _
```